Міністерство освіти і науки України

НТУУ «Київський політехнічний інститут»

Фізико-технічний інститут

Програмування

Комп’ютерний практикум №7

**Використання об’єктно-орієнтованого підходу для розробки програмного забезпечення**

Варіант №14

**Виконав:**

Студент 1 курсу ФТІ

Групи ФІ-92

Поночевний Назар Юрійович

**Перевірив:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Прийняв:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Використання об’єктно-орієнтованого підходу для розробки програмного забезпечення**

**Мета роботи:** засвоїти принципи об’єктно-орієнтованого проектування, включаючи аналіз предметної області та побудову фізичних і логічних моделей.

**Завдання:**

14. Розробити проект автопілота літака Boeing 737-900ER.

|  |  |
| --- | --- |
| **Актори** | **Сюжетна лінія** |
| Пілот | Пілот сідає у літак, виконує зліт літака і вмикає автопілот, який за допомогою системи керування підтримує необхідні показники літака під час польоту |
| Автопілот |
| Система керування |
| Літак Boeing 737-900ER |

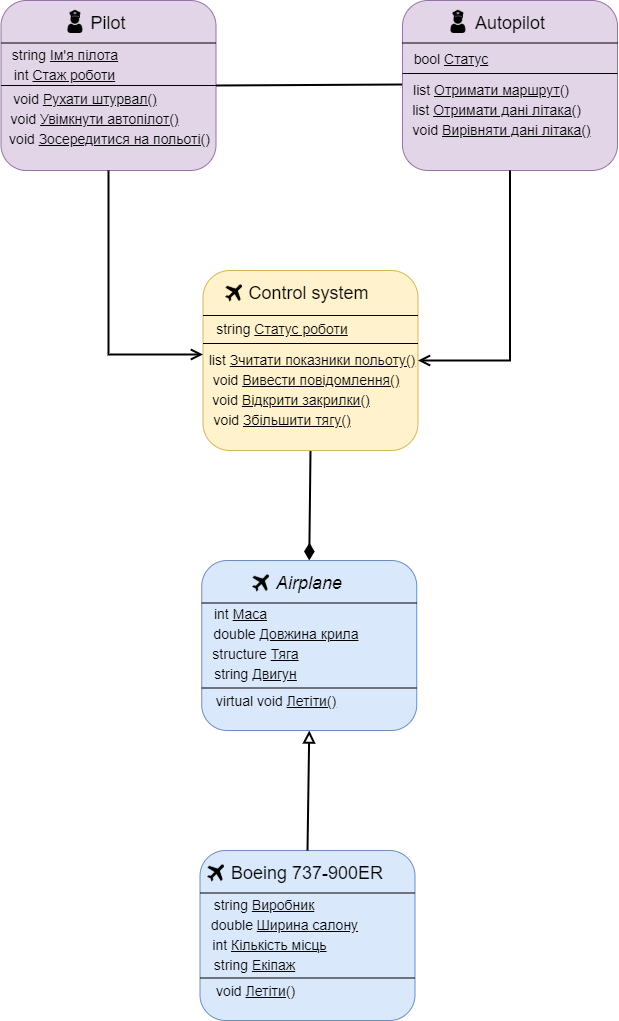
**Внутрішні / зовнішні методи**

|  |  |  |  |
| --- | --- | --- | --- |
| **Актор** | **Метод** | **Зовнішній** | **Внутрішній** |
| *Пілот* | Рухати штурвал | **✓** |  |
| Говорити з пасажирами | **✓** |  |
| Увімкнути автопілот | **✓** |  |
| Зосередитися на польоті |  | **✓** |
| *Автопілот* | Отримати маршрут | **✓** |  |
| Вирівняти параметри літака | **✓** |  |
| Провести калібрування |  | **✓** |
| *Система керування* | Зчитувати показники польоту | **✓** |  |
| Виводити повідомлення | **✓** |  |
| Відкрити закрилки | **✓** |  |
| Спустити шасі | **✓** |  |
| Збільшити тягу | **✓** |  |



**Use case diagram**

**Class diagram**



**Відношення між класами**

Клас Pilot:

* двостороння асоціація з класом Autopilot;
* використання класу Control system.

Клас Autopilot:

* двостороння асоціація з класом Pilot;
* використання класу Control system.

Клас Control system:

* композиція (строга агрегація) з класом Airplane (пояснення: літак стає некерованим без системи керування, тому система керування є необхідною частиною літака).

Клас Boeing 737-900ER:

* відношення успадкування від абстрактного класу Airplane.

**Код програми**

|  |  |  |
| --- | --- | --- |
| **pilot.py**   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84 | **from** **time** **import** sleep  **from** **random** **import** choice, randint  **class** **Pilot**:  focus\_level = **0**  **def** **\_\_init\_\_**(self, name="Vasya Pechkin", work\_experience=**0**):  self.name = name  self.work\_experience = work\_experience  **def** **\_\_str\_\_**(self):  **return** self.name  **@property**  **def** **work\_experience**(self):  **return** self.\_\_work\_experience  **@work\_experience**.setter  **def** **work\_experience**(self, work\_experience):  **if** work\_experience < **0**:  self.\_\_work\_experience = **0**  **elif** work\_experience > **100**:  self.\_\_work\_experience = **100**  **else**:  self.\_\_work\_experience = work\_experience  **@staticmethod**  **def** **move\_helm**(direction, control\_system):  control\_system.open\_flaps(direction)  **@staticmethod**  **def** **pull\_lever**(number, control\_system):  control\_system.change\_traction(number)  **@staticmethod**  **def** **enable\_autopilot**(control\_system, time, debug=**False**):  autopilot = Autopilot(control\_system)  autopilot.run(time, debug)  **return** autopilot.status  **def** **focus\_on\_flying**(self):  self.focus\_level += **10**  **class** **Autopilot**:  status = **0**  **def** **\_\_init\_\_**(self, control\_system):  self.control\_system = control\_system  **def** **run**(self, time, debug=**False**):  self.status = **1**  **for** \_ **in** range(time):  route = self.get\_route()  airplane\_data = self.get\_airplane\_data()  **if** debug:  print("**\n**Autopilot ask for the route from dispatcher...")  print("Route from dispatcher:", route)  print("Airplane traction:", airplane\_data['traction'])  print("Autopilot corrected airplane's traction")  self.correct\_airplane\_data(route, airplane\_data)  sleep(**1**)  self.status = **0**  **@staticmethod**  **def** **get\_route**():  # actually, we need to get the route from dispatcher,  # but we don't want to build a complex system, so we will generate route randomly  direction = choice(["Up", "Down", "Left", "Right"])  power = randint(**1000**, **5000**)  traction = [direction, power]  **return** traction  **def** **get\_airplane\_data**(self):  **return** self.control\_system.read\_flight\_indicators()  **def** **correct\_airplane\_data**(self, route, airplane\_data):  route\_direction, route\_power = route  direction, power = airplane\_data['traction']  **if** direction != route\_direction:  self.control\_system.open\_flaps(route\_direction)  **if** power != route\_power:  self.control\_system.change\_traction(route\_power) | |
| **airplane.py**   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | **from** **pilot** **import** Pilot  **class** **ControlSystem**:  status = "OK"  **def** **\_\_init\_\_**(self, airplane):  self.airplane = airplane  **def** **read\_flight\_indicators**(self):  **return** {  "traction": self.airplane.traction,  "position": self.airplane.position,  "weight": self.airplane.weight,  "engine": self.airplane.engine,  }  **@staticmethod**  **def** **display\_message**(message):  print(message)  **def** **open\_flaps**(self, direction):  self.airplane.traction[**0**] = direction  **def** **change\_traction**(self, number):  self.airplane.traction[**1**] = number  **class** **Airplane**:  weight = **1000**  wing\_length = **3.2**  traction = ["Down", **0**]  engine = "Pratt & Whitney JT5D"  position = {'x': **0**, 'y': **0**}  **def** **fly**(self):  **if** self.traction[**0**] == "Up":  self.position['y'] += self.traction[**1**] / self.weight  **if** self.traction[**0**] == "Down":  self.position['y'] -= self.traction[**1**] / self.weight  **if** self.traction[**0**] == "Left":  self.position['x'] -= self.traction[**1**] / self.weight  **if** self.traction[**0**] == "Right":  self.position['x'] += self.traction[**1**] / self.weight  **class** **Boeing737**(Airplane):  manufacturer = "Boeing"  \_\_interior\_width = **5.25**  seats\_number = **220**  weight = **5000**  wing\_length = **6.4**  engine = "Pratt & Whitney JT8D"  **def** **\_\_init\_\_**(self, crew=**None**):  **if** crew **is** **None**:  crew = [Pilot("Michi Kovalsky", **10**), Pilot("Kolya Stepanov", **7**), "Natalya Kolesnikova"]  self.crew = crew  **def** **fly**(self):  **if** self.crew[**0**].work\_experience > **5**:  **if** self.traction[**0**] == "Up" **and** self.traction[**1**] > **1000**:  self.position['y'] += self.traction[**1**] / self.weight  **if** self.traction[**0**] == "Down" **and** self.traction[**1**] > **1000**:  self.position['y'] -= self.traction[**1**] / self.weight  **if** self.traction[**0**] == "Left" **and** self.traction[**1**] > **1000**:  self.position['x'] -= self.traction[**1**] / self.weight  **if** self.traction[**0**] == "Right" **and** self.traction[**1**] > **1000**:  self.position['x'] += self.traction[**1**] / self.weight  **return** "OK"  **return** "Work experience of the main pilot is lower than 5 years. Change the main pilot." | |
| **main.py**   |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | **from** **time** **import** sleep  **from** **pilot** **import** Pilot  **from** **airplane** **import** ControlSystem, Boeing737  **def** **show\_pilot\_info**(pilot):  print("Pilot name:", pilot.name)  print("Work experience:", pilot.work\_experience)  print("Focus level:", pilot.focus\_level)  **def** **show\_airplane\_info**(airplane, show\_main=**False**):  print("Manufacturer:", airplane.manufacturer)  **if** **not** show\_main:  print("Seats number:", airplane.seats\_number)  print("Wing length:", airplane.wing\_length)  print("Engine:", airplane.engine)  print("Weight:", airplane.weight)  print("Crew: {}, {}, {}".format(airplane.crew[**0**], airplane.crew[**1**], airplane.crew[**2**]))  print("Traction:", airplane.traction)  print("Position:", airplane.position)  **def** **main**():  crew = [Pilot("Michi Kovalsky", **10**), Pilot("Kolya Stepanov", **7**), "Natalya Kolesnikova"]  boeing737\_airplane = Boeing737(crew)  control\_system = ControlSystem(boeing737\_airplane)  main\_pilot = crew[**0**]  show\_pilot\_info(main\_pilot)  print("**\n**Main pilot focusing on fly...**\n**")  main\_pilot.focus\_on\_flying()  show\_pilot\_info(main\_pilot)  print("**\n**Airplane status:")  show\_airplane\_info(boeing737\_airplane)  print("**\n**Main pilot moving helm 'Up' and pull lever to '6000'...")  main\_pilot.move\_helm("Up", control\_system)  main\_pilot.pull\_lever(**6000**, control\_system)  print("**\n**Airplane start flying...")  **for** \_ **in** range(**10**):  print("**\n**Airplane status (main):")  show\_airplane\_info(boeing737\_airplane, show\_main=**True**)  boeing737\_airplane.fly()  sleep(**1**)  print("**\n**Main pilot enabling autopilot...")  autopilot\_status = main\_pilot.enable\_autopilot(control\_system, **10**, debug=**True**)  print("**\n**Autopilot finished work with status", autopilot\_status)  **if** \_\_name\_\_ == '\_\_main\_\_':  main() | |
| **Output:**  Pilot name: Michi Kovalsky  Work experience: 10  Focus level: 0  Main pilot focusing on fly...  Pilot name: Michi Kovalsky  Work experience: 10  Focus level: 10  Airplane status:  Manufacturer: Boeing  Seats number: 220  Wing length: 6.4  Engine: Pratt & Whitney JT8D  Weight: 5000  Crew: Michi Kovalsky, Kolya Stepanov, Natalya Kolesnikova  Traction: ['Down', 0]  Position: {'x': 0, 'y': 0}  Main pilot moving helm 'Up' and pull lever to '6000'...  Airplane start flying...  Airplane status (main):  Manufacturer: Boeing  Traction: ['Up', 6000]  Position: {'x': 0, 'y': 0}  Airplane status (main):  Manufacturer: Boeing  Traction: ['Up', 6000]  Position: {'x': 0, 'y': 1.2}  Airplane status (main):  Manufacturer: Boeing  Traction: ['Up', 6000]  Position: {'x': 0, 'y': 2.4}  …  Main pilot enabling autopilot...  Autopilot ask for the route from dispatcher...  Route from dispatcher: ['Right', 1755]  Airplane traction: ['Up', 6000]  Autopilot corrected airplane's traction  Autopilot ask for the route from dispatcher...  Route from dispatcher: ['Up', 1526]  Airplane traction: ['Right', 1755]  Autopilot corrected airplane's traction  Autopilot ask for the route from dispatcher...  Route from dispatcher: ['Down', 1120]  Airplane traction: ['Up', 1526]  Autopilot corrected airplane's traction  …  Autopilot finished work with status 0 |
| **Screenshot** |